Politecnico di Torino

Digital systems electronics
A.A. 2018/2019

Prof. G. Masera

Lab 05

16 apr 2019

Berchialla Luca	236032
Laurasi Gjergji	238259
Mattei Andrea	233755
Lombardo Domenico Maria	233959

1 One-Hot Finite state machine

The FSM from the STG diagram given in *figure* 1 was implemented using a One-Hot state assigement shown in *figure* 2 with the circuit shown in *figure* 3.

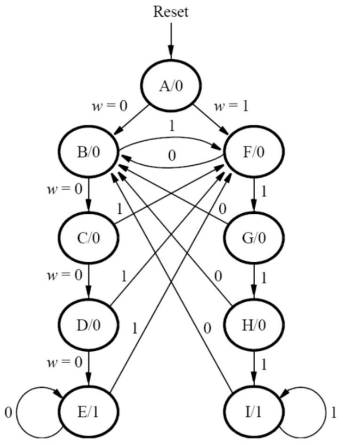

Figure 1: State diagram of the FSM

Name	State Code $y_8y_7y_6y_5y_4y_3y_2y_1y_0$
A	000000001
B	000000010
C	000000100
D	000001000
E	000010000
F	000100000
G	001000000
H	010000000
I	100000000

Figure 2: One-Hot code

The correct behavior of the circuit has been verified by means of a testbench. The result of the simulation are shown in *figure* 4.

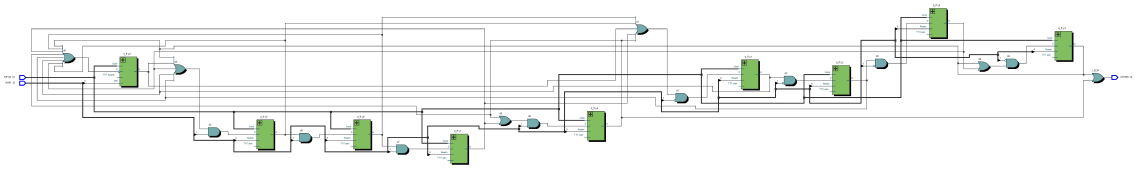

Figure 3: RTL View

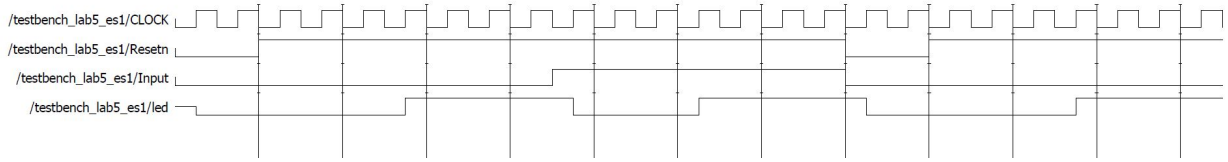

Figure 4: Testbench waveforms

2 Modified One-Hot FSM

In this section we had to implement the same FSM of the previous section, but with another One-Hot code shown in *figure* 5. The value of this state encoding is that the reset state is coded as 000000000. This excludes the need of a set port for the Flip Flops in the circuit that now have only a reset port.
The Circuit that implements this FSM with this state encoding is shown in *figure* 6 and is very similar to the one of the previous section. An inverter at the output of the $y0$ Flip Flop was added while all the other connections are unchanged.

Name	State Code $y_8y_7y_6y_5y_4y_3y_2y_1y_0$
A	000000000
B	000000011
C	000000101
D	000001001
E	000010001
F	000100001
G	001000001
H	010000001
I	100000001

Figure 5: Modified One-Hot code

3 Two-process FSM

The task of this part is to provide an alternative implementation of the previous FSM. This implementation is composed of 3 processes each describing one of the 3 fundamental components of a generic FSM (2 combinational and 1 sequential): the first combinational circuit determines the future state based on the current state and inputs, the other combinational circuit manages the output according to the current state and the state flip-flops that store the current state.
A single VHDL source file contains all the circuit. The inputs of the circuit are the first 2 switches (SW0 for the synchronous reset and SW1 for real input) and the button KEY0 used for simulating, while the sole single bit output drives the LEDR0 LED. Using this architecture, Quartus Prime during compilation successfully recognizes that the circuit is a state machine and generates the relative state diagram, which corresponds to the desired one.
The circuit was tested using a testbench that generates all the possible state transitions. The same testbench tested also the synchronous reset mechanism.

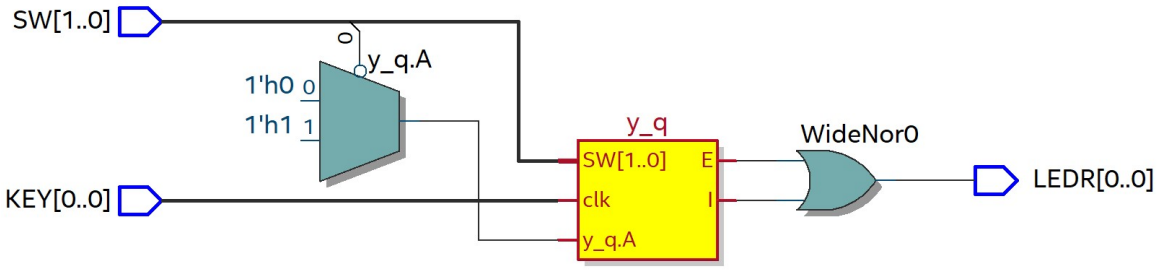

Figure 6: The RTL generated of the state machine of part 3

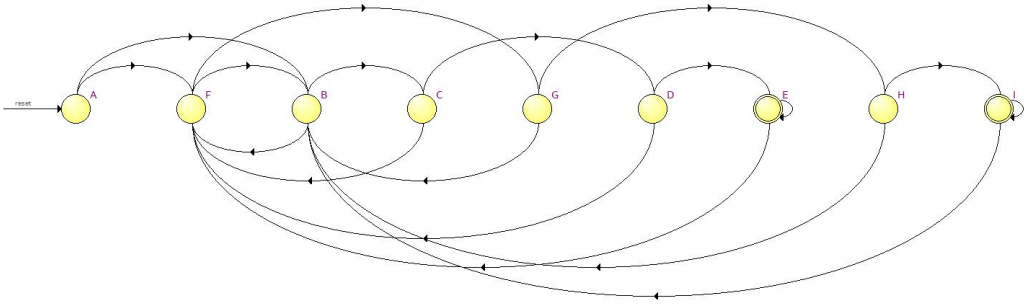

Figure 7: The state diagram of the state machine of part 3

4 - "HELLO" FSM

In this section a circuit that scrolls the word "HELLO" over the display has been implemented. The image below shows the architecture of the circuit generated using a VHDL description:

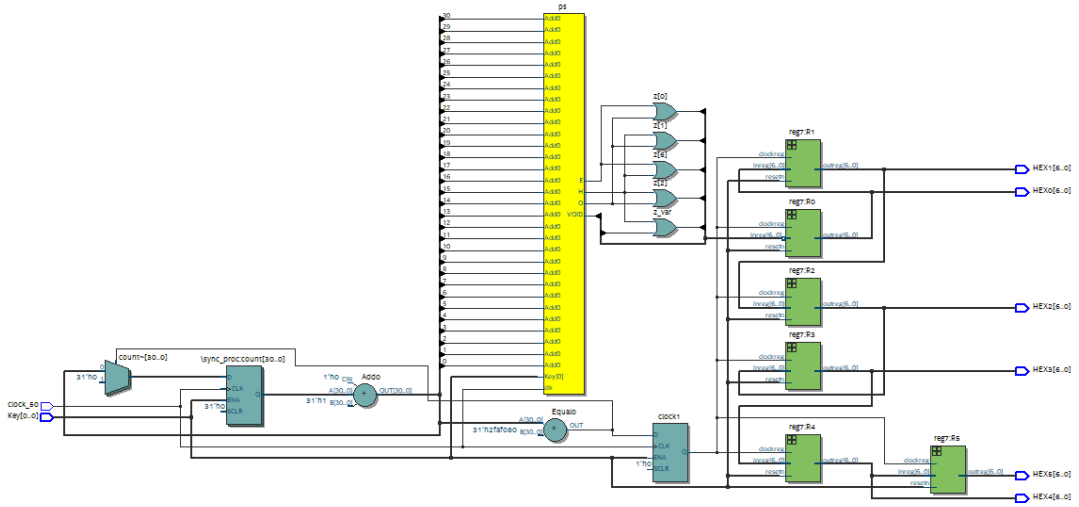

Figure 8: Implemented architecture

The circuit uses six 7-bit registers connected in a pipeline fashion. Each of them directly drives a 7-segment display.

The FSM controls the pipeline by inserting the characters (H,E,L,L,O) into the

first 7-bit register. Every second the letters scroll from right to left, once the cycle is completed (i.e. The 'O' letter reach the leftmost display) the FSM starts the process again in an infinite loop. The state diagram of the implemented FSM is shown below:

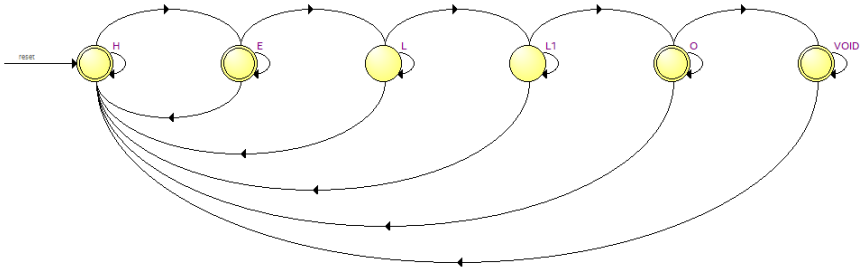

Figure 9: State Diagram

The 6 7-bit resisters have been implemented using a behavioral approach. The testbench shown below has been performed to check their functionalities:

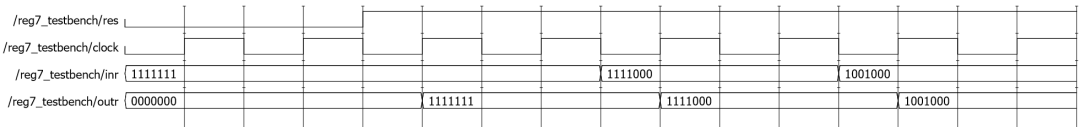

Figure 10: 7-bit register testbench

Finally a testbench for the entire design has also been implemented, showing the correct behavior of the circuit.

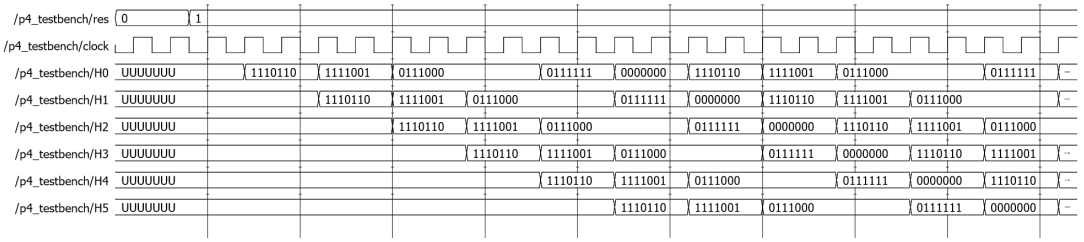

Figure 11: "HELLO" FSM testbench

Note that to keep the simulation fast enough the letter scrolls every 2 clock cycles.